SPHERE
NEWSLETTER

JUNE 1979 *VOLUME IV ISSUE 6*

EDITORS:
ROGER J. SPOTT
JEFF BROWNSTEIN

SOFTWARE

HARDWARE

EDITOR'S MAILBAG

RENEWALS
******VERY IMPORTANT*******
*********RENEW NOW*********

See page 27

PLEASE SEND TYPED MATERIAL FOR NEXT ISSUE TO:

ROGER J. SPOTT
13975 CONNECTICUT AVENUE
WHEATON, MARYLAND 20906

FROM THE EDITOR'S DESK

Needed: Schematic and info for SINGER FRIEDEN HSP-30 Serial printer(Mod14325) Please contact Larry Sambuco, 22 Fredrick Dr. Poughkeepsie, NY. 12603

Correction: The last fix for CSS DEF FN (see page 16 last issue) should read: 18FE BD0313

New fix: If you are not able to display the stack on your CRT1 or 1A board Larry Sambuco has a correction for the board. E7 Pin 1 should be clocked only by a jumper coming from E19 pin 12. Larry also put a 470 Pf. capacitor from E7 pin 8 to ground. Some Spheres have no problem in displaying the stack on the screen so check first before getting out the soldering iron.

64 Character screen fix for Chuck Crayne OS1 Disk Operating System Editor:

Address for V. 1.0	V 2.0	Change
080E	0CB3	E020 to E040
0815	0CBA	E200 to E400
08DE	07DE	20 to 40
08E6	0D86	20 to 40
08F8	0D98	20 to 40
0901	0DA1	E200 to E400
091D	0DBD	E0 to C0
0978	0E1D	E1E0 to E3C0
0992	0E37	E020 to E040
0994	0E3A	20 to 40
0999	0E3F	E1E0 to E3C0
09A6	0E4C	E1DF to E3BF
09AB	0E50	20 to 40
09B0	0E56	E01F to E03F
09BA	0E60	E03F to E07F
09F5	0EA2	E020 to E040
0A1A	0EC9	E01A to E03A
0A27	0ED4	E020 to E040

In addition I have written a utility which is used with the OS1 version 1.0 but which could be also converted to version 2.0 . It sits up in ROM and performs the following:

ALLOCATE (to directory)
KILL (remove filename from directory)
DISPLAY (display the directory)
EXIT (return to the calling program).

This allows Basic, Forth, Pascal and other programs to have a command which gives control to the user of the file directory of that disk. The job of saving and loading program or data files is left to the calling program because each one will necessarily use a format which is unique.

Jeff

RRR
EEE
NNN
EE
WW
AA
LL
SS.

YOU MUST RE-SUBSCRIBE NOW (a) 12.50

There is no money in the bank account to finance another issue. Rising printing costs and postage have eliminated our reserves.

WE ALSO WILL NEED MORE MATERIAL TO CONTINUE THIS NEWSLETTER. EVEN IF IT IS NOT TYPED____>SEND IT!!!!!!!!!

I HAVE BEEN USING THE BETA DIGITAL TAPE DRIVE BY MECA ON MY SPHERE FOR APPROXIMATELY A YEAR. IN THAT TIME I HAVE FOUND IT TO BE AN EXTREMELY RELIABLE DRIVE. I WENT TO THIS DRIVE BECAUSE IT IS EQUIPPED WITH AN INTELLIGENT CONTROLLER (8035 WITH 2K ROM OPER SYS) WHICH ALLOWS IT TO BE EASILY "IO MAPPED" INTO A PIA.

I AM USING STANDARD TDK (AD SERIES) CASSETTE TAPES INSTEAD OF THE RECOMMENDED DIGITAL TAPES WHICH ARE CONSIDERABLY MORE EXPENSIVE. CHEAPER TAPES CAN BE USED AND WILL WORK, BUT THE HIGH SPEED SEARCH OF 100 INCHES/SECOND DEMANDS THE BETTER MECHANICS I FOUND IN THE TDK SERIES OF TAPES.

HERE ARE SOME OF THE HIGH POINTS OF THE DRIVE:

1. 512K BYTES/60 MIN CASSETTE.
2. DATA FORMAT IS A 256 BYTE BLOCK PRECEEDED BY AN ID.
3. ANY RECORD CAN BE RE-WRITTEN WITHOUT DISTURBING THE ONE BEFORE OR AFTER IT. AS A MATTER OF FACT, BECAUSE OF THE DISK DRIVE APPEARANCE TO SOFTWARE, I SUPPOSE YOU COULD EVEN RUN IT UNDER A DOS, OF COURSE AT A GREATLY REDUCED SPEED.
4. THE FIRST DRIVE WITH THE INTEGRAL CONTROLLER COSTS $400.00 AND EXTRA DRIVES (UP TO 3) COST AN ADDITIONAL $270.00.

   NOTE: THESE PRICES MAY HAVE CHANGED SINCE MY PURCHASE.
5. HIGH LEVEL COMMANDS ARE USED TO ACCESS AND CHANGE DATA, CONTROL THE DRIVE(S), ETC.
6. USES ONLY 1 PIA CHIP (BOTH A AND B SIDES).
7. EACH TAPE USES AN UPPER AND A LOWER TRACK, SO NO REVERSAL OF THE CASSETTE IS NECESSARY.
8. OPERATION OF THE DRIVE IS 100 PERCENT SOFTWARE CONTROLLED.

9. HIGH SPEED SEARCH CAPABILITY BY COUNTING TAKE UP REEL REVOLUTIONS USING A TACHOMETER. THIS MEANS THAT IF YOU HAVE JUST READ BLOCK 258 AND THEN WANT TO READ BLOCK 017, THE BETA WILL UNLOAD IT'S HEAD, AND HIGH SPEED REWIND TILL JUST BEFORE BLOCK 017. IT WILL THEN LOAD THE HEAD AND PROCEED TO READ THE BLOCK.

10. FOUR MOTORS = CAPSTAN, LEFT REEL, RIGHT REEL, AND HEAD LOAD/UNLOAD.

HERE'S THE TAPE FORMAT.....

GAP - ID - GAP - 256 BYTE DATA BLOCK - GAP - ID - GAP - 256 BYTE DATA BLOCK - ETC.

THE WRITE AND READ DATA RATES ARE 500 BYTES/SEC WHICH WILL NOT CAUSE ANY PROBLEMS WITH THE SPHERE. YOU CAN ALSO USE THE ON BOARD BUFFER IN THE BETA TO DO A "BACK GROUND" TAPE COPY WITHOUT THE NEED TO FIRST READ DATA FROM ONE DRIVE INTO THE SPHERE, AND THEN SEND IT BACK TO WRITE IT ON ANOTHER DRIVE. IN THIS MODE THE ONLY INTERFACE ACTIVITY WILL BE TO ISSUE A COMMAND TO READ FROM ONE DRIVE, CK ENDING STATUS, THEN WRITE THE BUFFER CONTENTS ONTO ANOTHER DRIVE. THE DATA NEED NEVER GO INTO SPHERE MEMORY. PRETTY NEAT, HUH ? YOU CAN ALSO DO "PEEKS" AND "POKES" DIRECTLY INTO THE BETA BUFFER.

THE TOTAL AGGREGATE DATA IS ABOUT 385 BYTES/SEC WHEN READING MULTIPLE BLOCKS OF DATA BECAUSE OF THE GAPS AND ID'S. MECA ALSO HAS A FASTER DRIVE AVAILABLE BUT THAT'S MORE BUCKS AND MEANS CHANGING THE ORIGINAL TAPE DECKS FOR FASTER TAPE DECKS (1000 BYTES/SEC).

II

INCLUDED WITH THIS BRIEF INTRODUCTION OF THE BETA IS MY "OPERATING SYSTEM" WHICH VERY CLOSELY DUPLICATES SPHERE CASSETTES. ALSO, I PAD IN THE EVENT YOU ARE NOT WRITING OR READING AN EXACT MULTIPLE OF 256 BYTES, WHICH HAPPENS QUITE NORMALLY. THIS MEANS IF YOU WANT TO SAVE A PROGRAM AT $0200 TO $02C6 THE CODE WILL PAD THE REMAINING BYTES SO A FULL BLOCK IS WRITTEN. AS THE BYTE CT IS RECORDED, YOU ONLY HAVE TO SPECIFY THE START ADDRESS WHEN DOING A READ AND THE ENDING DATA ADDRESS WILL BE BUILT AT $3E AND $3F. ALSO, WHEN THE ABOVE EXAMPLE RECORD IS READ BACK, ONLY THE CONTENTS OF $0200 TO $02C6 WILL BE READ INTO, AND THE EXTRA BYTES DISCARDED AGAIN TO COMPLETE A 256 BYTE BLOCK IN ORDER TO KEEP THE BETA HAPPY.

BETA PARAMETER USAGE FOLLOWS:

$33,34 = TBBB WHERE T IS A TRACK NUMBER FROM 0 TO 7 AND B IS A BLOCK NUMBER FROM 000 TO 999.

NOTE: TRACK 0 AND 1 IS ON DRIVE 0, TRACK 2 AND 3 IS ON DRIVE 1, ETC.

$38,39 = NOT USED

$3C,3D = STARTING DATA ADDRESS

$3E,3F = ENDING DATA ADDRESS

ROM ENTRY POINT DESCRIPTIONS

THE FOLLOWING LABELS WILL RETURN TO DEBUG WHEN FINISHED...

LABEL	FUNCTION DESCRIPTION	$0033,34	$003C,3D	$003E,3F
BINIT	INIT PIA	N/A	N/A	N/A
BWRITE	WRITE FROM MEMORY AND VERIFY	TBBB	BEG ADR	END ADR
BREAD	READ INTO MEMORY	TBBB	BEG ADR	CALCULATED
BINREW	INIT PIA AND REWIND UNIT SEL'D BY $0033 (T)	TXXX	N/A	N/A
REWIND	REWINDS UNIT SEL'D BY $0033 (T)	TXXX		

THE FOLLOWING LABLES ARE SUBROUTINE ENTRY POINTS...

LABEL	FUNCTION DESCRIPTION	$0033,34	$003C,3D	$003E,3F
WRITE2	WRITE FROM MEMORY AND VERIFY	TBBB	BEG ADR	END ADR
READ2	READ INTO MEMORY	TBBB	BEG ADR	CALCULATED
VERIF2	VERIFY DATA (NO DATA XFER)	TBBB	BEG ADR	END ADR
RWIND	REWINDS UNIT SEL'D BY $0033 (T)	TXXX	N/A	N/A
INIT	INIT PIA	N/A	N/A	N/A

XXX USED ABOVE = DON'T CARE

NOTE 1: $0009 IS SET TO NON-ZERO IF A READ, VERIFY, OR TAPE PROTECTED ERROR OCCURS AND CLEARED OTHERWISE, SO IT IS TESTABLE JUST AS IF YOU WERE USING THE SPHERE CASSETTES. ALSO, IF $003A IS ZERO, NO PRINTING WILL OCCUR ON THE CRT, OTHERWISE THE USUAL WRITE AND READ POSITIONS IN THE UPPER RIGHT HAND OF THE CRT WILL BE UPDATED AS DATA IS READ OR WRITTEN TO THE CONTROLLER. I DO AN ENQUIRY COMMAND AT THE COMPLETION OF EVERY COMMAND AND WILL PRINT IT AT THE CURRENT CURSOR POSITION UNLESS $003A IS ZERO.

NOTE 2: THE "BOOT" AT $F818 DOES A PIA INIT, REWINDS THE DRIVE SELECTED BY THE LEFT NIBBLE OF $0033(T), READS THE SPECIFIED BLK(BBB) INTO THE ADDRESS SPECIFIED AT $003C, TESTS $0009 FOR AN ERROR, AND JUMPS TO THE ADDRESS AT $003C FOR EXECUTION IF NO ERROR, OR RETURNS TO DEBUG.

NOTE 3: PLEASE NOTE THAT THERE ARE THINGS IN THIS MONITOR THAT SHOULD BE OBVIOUSLY "CLEANED UP" BUT WEREN'T BECAUSE OF LACK OF TIME. THIS MONITOR WILL WORK QUITE NICELY, BUT IF YOU SEE SOMETHING YOU DON'T CARE FOR, FEEL FREE TO CHANGE IT.

SOME PROBLEMS I CAN THINK OF ARE...

1. IF $003A IS ZERO AND AN ERROR OCCURS, I DO AN ENQUIRY COMMAND TO THE CONTROLLER AND PRINT IT ANYWAY. IF I HAD FOLLOWED THE USE OF THE PRINT/NO-PRINT FLAG ($3A) STRICTLY, I SHOULD NEVER PRINT UNLESS THE FLAG IS NON-ZERO.
2. THE OCCURRENCE OF AN "ECHO" ERROR WILL ALWAYS GO TO THE MONITOR. I SHOULD RE-CODE THAT AREA TO ISSUE AN ABORT SEQUENCE TO THE BETA, SET $0009 NON-ZERO, AND RETURN TO THE USER OR DEBUG DEPENDING ON THE ENTRY POINT USED.

```
003C            BEGADR EQU  $3C
0007            BELL   EQU  $07
0033            BLKNO  EQU  $33
F040            BRDATA EQU  $F040
F041            BRDCTL EQU  $F041
F043            BWRCTL EQU  $F043
F042            BWRDAT EQU  $F042
0018            CLRCMD EQU  $18
000D            CR     EQU  $0D
FE4F            DEBUG  EQU  $FE4F
FD88            DSPRD  EQU  $FD88
FD6E            DSPWR  EQU  $FD6E
003E            ENDADR EQU  $3E
050D            ENDCMD EQU  $050D
E00F            ERRIND EQU  $E00F
001B            ESC    EQU  $1B
000A            LF     EQU  $0A
E040            LINE2  EQU  $E040
F000            KBDATA EQU  $F000
F001            KBDRCD EQU  $F001
003A            NOPRNT EQU  $3A
FCAD            PUTCHR EQU  $FCAD
0012            RDCMD  EQU  $12
1A0D            REWCMD EQU  $1A0D
0009            STATUS EQU  $09
BFF9            SWIFLG EQU  $BFF9
0014            TRKCMD EQU  $14
BFFA            UIRVEC EQU  $BFFA
BFFD            UNMVEC EQU  $DFFD
BFF6            USWVEC EQU  $BFF6
0016            VERCMD EQU  $16
0000            WKCMD  EQU  $0000
0017            WRCMD  EQU  $17
                *
F810             ORG  $F810
                *
                * BOOT LOAD FROM BETA
                *
F810 BD F866     JSR  INITREW
F813 BD FA1D     JSR  READ
F816 DE 3C       LDX  BEGADR
F818 96 09       LDAA STATUS  RD ERROR ?
F81A 26 02       BNE  AARGH
F81C 6E 00       JMP  0,X  GOTO USER PGM
F81E 7E FE4F    AARGH JMP  DEBUG
                *
                * CK MONITOR OR USER SWI
                * IF SWIFLG=0 THEN USER
                * IF SWIFLG<>0 THEN MON
                *
F821 7D BFF9    CKSWI  TST  SWIFLG
F824 27 03       BEQ  USWI
F826 7E FE4F     JMP  DEBUG
```


5

```
                *
                * GOTO USER SWI HANDLER
                *
F829 7E BFF6    USWI JMP USWVEC
                *
F840             ORG  $F840
                *
F840 BD F86A    BINIT  JSR  INIT
F843 7E FE4F    DEBG   JMP  DEBUG
F846 BD F850    BWRITE JSR  WRITE2
F849 20 F8       BRA  DEBG
F84B BD FA1D    BRDMD  JSR  READ
F84E 20 F3       BRA  DEBG
F850 BD F938    WRITE2 JSR  WRITE
F853 96 09       LDAA STATUS
F855 27 00       BEQ  VERIF2
F857 39          RTS
F858 8D 0C      BINREW BSR  INITREW
F85A 20 E7       BRA  DEBG
F85C 7E FA1D    READ2  JMP  READ
F85F 7E FA8C    VERIF2 JMP  VERIFY
F862 8D 2A      REWIND BSR  RWIND
F864 20 DD       BRA  DEBG
F866 8D 02      INITREW BSR  INIT
F868 20 24       BRA  RWIND
                *
F86A 4F         INIT CLRA
F86B B7 F041     STAA BRDCTL SEL
F86E B7 F043     STAA BWRCTL   DDR'S
F871 B7 F040     STAA BRDATA  A=INP
F874 43          COMA
F875 B7 F042     STAA BWRDATA B=OUT
F878 86 2C       LDAA #$2C
F87A B7 F041     STAA BRDCTL A=+IRQ STB'D
F87D 7D F040     TST  BRDATA  RES A IRQ
F880 86 2E       LDAA #$2E
F882 B7 F043     STAA BWRCTL B=-IRQ STB'D
F885 8D 13       BSR  WTELF
F887 86 18       LDAA #CLRCMD
F889 B7 F042     STAA BWRDAT
F88C 20 0C       BRA  WTELF
                *
                * SEL TRACK USING LEFT
                * NIBBLE OF 'BLKNO' AND
                * DO A REWIND.
                *
F88E BD F909    RWIND JSR  SELTRK
F891 CE 1A0D     LDX  #REWCMD
F894 DF 00       STX  WKCMD
F896 8D 13       BSR  SNDCMD
F898 20 3C       BRA  ENDOP
                *
                * LOOP FOR 4 MS OR UNTIL
                * BETA SENDS A LINE FD.
```

```
                    *
F89A 5F             WTELF  CLRB
F89B 5A             CTDOWN DECB
F89C 27 0C           BEQ   RTN1
F89E 7D F041         TST   BRDCTL
F8A1 2A F8           BPL   CTDOWN
F8A3 B6 F040         LDAA  BRDATA
F8A6 81 0A           CMPA  #LF
F8A8 26 F1           BNE   CTDOWN
F8AA 39             RTN1   RTS
                    *
                    * SEND CMD TO BETA AND
                    * CK FOR PROPER ECHO.
                    *
F8AB CE 0000        SNDCMD LDX  #WKCMD
F8AE A6 00           LDAA 0,X
F8B0 8D 67           BSR   SNDRCV  SEND CMD
F8B2 BD F930         JSR   RDBETA  READ ECHO
F8B5 84 BF           ANDA #$BF     IGNORE BIT6
F8B7 A1 00           CMPA 0,X      ECHO OK ?
F8B9 27 0C           BEQ   NXTBYT
                    *
F8BB 31             ERRE    INS
F8BC 31              INS
F8BD C6 45           LDAB #'E
F8BF D7 09          ERRW    STAB STATUS
F8C1 F7 E00F         STAB ERRIND
F8C4 7E FE4F         JMP   DEBUG  DIE!DIE!DIE!
                    *
F8C7 08             NXTBYT INX
F8C8 A6 00           LDAA 0,X
F8CA 8D 4D           BSR   SNDRCV
F8CC A1 00           CMPA 0,X      ECHO OK ?
F8CE 26 EB           BNE   ERRE
F8D0 81 0D           CMPA #CR
F8D2 26 F3           BNE   NXTBYT
F8D4 20 C4           BRA   WTELF
                    *
                    * GOOD END = 07,0D,0A
                    * BAD = 07,ERR,07,0D,0A
                    *
F8D6 8D 58          ENDOP  BSR   RDBETA
F8D8 81 07          CKBELL CMPA #BELL
F8DA 26 FA           BNE   ENDOP
F8DC 5F              CLRB
F8DD 8D 51           BSR   RDBETA
F8DF 81 0D           CMPA #CR
F8E1 27 03           BEQ   RDLF
F8E3 53             HAVENR COMB
F8E4 97 09           STAA STATUS
F8E6 8D 48          RDLF   BSR   RDBETA
F8E8 81 0A           CMPA #LF
F8EA 26 FA           BNE   RDLF
F8EC 5D              TSTB
```



```
F8ED 26 08           BNE   DOENQ
F8EF 7F 0009         CLR   STATUS
F8F2 96 3A           LDAA NOPRNT
F8F4 26 01           BNE   DOENQ
F8F6 39              RTS
                    *
                    * PRINT CURRENT DRIVE
                    * STATUS ON CRT.
                    *
F8F7 86 0D          DOENQ  LDAA #CR
F8F9 BD FCA0         JSR   PUTCHR
F8FC CE 0500         LDX   #ENQCMD
F8FF DF 00           STX   WKCMD
F901 8D A8           BSR   SNDCMD
F903 8D 28          ENQ    BSR   RDBETA
F905 81 07           CMPA #BELL
F907 27 FA           BEQ   ENQ
F909 81 0A           CMPA #LF
F90B 27 9D           BEQ   RTN1
F90D BD FCA0         JSR   PUTCHR
F910 20 F1           BRA   ENQ
                    *
F912 97 09          ERROR   STAA STATUS
F914 B7 E00F         STAA ERRIND
F917 20 BD           BRA   ENDOP
                    *
                    * NOTE: IF ESC CHAR IS
                    * NORMAL DATA, IT IS
                    * SENT TWICE.
                    *
F919 8D 07          SNDRCV BSR   WRBETA
F91B 8D 13           BSR   RDBETA
F91D 81 1B           CMPA #ESC
F91F 27 01           BEQ   WRBETA
F921 39              RTS
                    *
                    * WRITE A BYTE TO BETA
                    *
F922 7D F043        WRBETA TST   BWRCTL
F925 2A FB           BPL   WRBETA
F927 7D F042         TST   BWRDAT  RESET IRQ
F92A B7 F042         STAA BWRDAT
F92D 7E FB6E         JMP   DSPWR
                    *
                    * READ A BYTE FROM BETA
                    *
F930 7D F041        RDBETA TST   BRDCTL
F933 2A FB           BPL   RDBETA
F935 B6 F040         LDAA BRDATA
F938 7E FB88         JMP   DSPRD
                    *
F93B C6 17          WRITE  LDAB #WRCMD
F93D 8D 3D           BSR   SETUP
F93F 96 3F           LDAA ENDADR+1
```

```
F941 D6 3E      LDAB ENDADR
F943 90 3D      SUBA BEGADR+1  BA = BYTE
F945 D2 3C      SBCB BEGADR      COUNT.
F947 37         PSHB
F948 16         TAB
F949 8D CE      BSR  SNDRCV WR LO CT BYT
F94B 10         SBA
F94C 33         PULB
F94D 4D         TSTA  ECHO OK ?
F94E 27 03      BEQ  WCT2
F950 7E F8D8  WCTERR JMP  CKBELL
                *
F953 17         WCT2  TBA
F954 8D C3      BSR  SNDRCV WR HI CT BYT
F956 11         CBA   ECHO OK ?
F957 26 F7      BNE  WCTERR
F959 C6 FE      LDAB #256-2 B=BYT CTR
F95B DE 3C      LDX  BEGADR
F95D A6 00      WR1  LDAA 0,X
F95F 8D B8      BSR  SNDRCV
F961 A1 00      CMPA 0,X  ECHO OK ?
F963 27 03      BEQ  WR2
F965 7E F8D8    JMP  CKBELL
F968 5A         WR2 DECB  UPDATE BYT CTR
F969 9C 3E      CPX  ENDADR
F96B 27 03      BEQ  CKWPAD
F96D 08         INX
F96E 20 ED      BRA  WR1
                *
                * IF BC>0 THEN WRITE 0'S
                * TO FINISH A 256 BYTE
                * BLOCK.
                *
F970 5D         CKWPAD TSTB
F971 27 06      BEQ  NOWPAD
F973 4F         WRPAD  CLRA
F974 8D A3      BSR  SNDRCV
F976 5A         DECB
F977 26 FA      BNE  WRPAD
F979 7E F8D6  NOWPAD JMP  ENDOP
                *
                * SEL TRACK, STORE CMD,
                * BLOCK CT, ADDITIONAL
                * BLOCK CT, AND SEND
                * ALL TO BETA.
                *
F97C 37         SETUP  PSHB  SAVE CMD
F97D 8D 0A      BSR  SELTRK
F97F 33         PULB
F980 D7 00      STAB WKCMD
F982 8D 24      BSR  SETBLK
F984 8D 45      BSR  SETCT
F986 7E F8AB    JMP  SNDCMD
                *
```


2

```
                * SELECT TRACK USING LEFT
                * NIBBLE OF 'BLKNO'.
                *
F989 96 33      SELTRK LDAA BLKNO
F98B 44          LSRA
F98C 44          LSRA
F98D 44          LSRA
F98E 44          LSRA
F98F 84 07       ANDA #7
F991 8A 30       ORAA #$30
F993 97 01       STAA WKCMD+1
F995 C6 14       LDAB #TRKCMD
F997 D7 00       STAB WKCMD
F999 CC 0D       LDAD #CR
F99C D7 02       STAB WKCMD+2
F99E BD F8AB     JSR  SNDCMD
                *
F9A0 BD F930    WTELF2 JSR  RDBETA
F9A3 81 0A       CMPA #LF
F9A5 26 F9       BNE  WTELF2
F9A7 39          RTS
                *
                * 'BLKNO' FORMAT = TBBB,
                * WHERE T IS TRK, AND B
                * IS STARTING BLOCK.
                * AT END OF THIS SUB,
                * 'WKCMD' =
                * CMD B B B , X X X X
                *
F9A8 CE 0000    SETBLK LDX  #WKCMD
F9AB D6 33       LDAB BLKNO
F9AD 8D 17       BSR  ASCRT
F9AF E7 01       STAB 1,X
F9B1 D6 34       LDAB BLKNO+1
F9B3 8D 00       BSR  ASCLFT
F9B5 E7 02       STAB 2,X
F9B7 D6 34       LDAB BLKNO+1
F9B9 8D 08       BSR  ASCRT
F9BB E7 03       STAB 3,X
F9BD C6 2C       LDAB #',
F9BF E7 04       STAB 4,X
F9C1 39          RTS
F9C2 54         ASCLFT LSRB
F9C3 54          LSRB
F9C4 54          LSRB
F9C5 54          LSRB
F9C6 C4 0F      ASCRT  ANDB #$0F
F9C8 CA 30       ORAB #$30
F9CA 39          RTS
                *
                * AT END OF THIS SUB,
                * 'WKCMD' =
                * CMD B B B , C C C $0D
                *
```

```
F9CD CE 0000  SETCT  LDX  #WKCMD
F9CE A6 00     LDAA 0,X
F9D0 81 12     CMPA #RDCMD
F9D2 27 38     BEQ  MAXCT
F9D4 D6 3E     LDAB ENDADR
F9D6 96 3F     LDAA ENDADR+1
F9D8 90 3D     SUBA BEGADR+1
F9DA D2 3C     SBCB BEGADR
F9DC 8B 02     ADDA #2
F9DE C9 00     ADCB #0  B=EXTRA BLK CT
F9E0 6F 06     CLR  6,X
F9E2 6F 07     CLR  7,X
F9E4 5D        TSTB
F9E5 27 11     BEQ  CT2ASC
               *
               * CONVERT BLK CT IN B TO
               * PACKED DECIMAL IN
               * WKCMD+6,7.
               *
F9E7 A6 07     CT2DEC LDAA 7,X
F9E9 8B 01      ADDA #1
F9EB 19         DAA
F9EC A7 07      STAA 7,X
F9EE A6 06      LDAA 6,X
F9F0 89 00      ADCA #0
F9F2 19         DAA
F9F3 A7 06      STAA 6,X
F9F5 5A         DECB
F9F6 26 EF      BNE  CT2DEC
               *
               * CONV PACKED DECIMAL IN
               * WKCMD+6,7 TO ASCII
               * IN WKCMD+5,6,7.
               *
F9F8 E6 06     CT2ASC LDAB 6,X
F9FA 8D CA      BSR  ASCRT
F9FC E7 05      STAB 5,X
F9FE E6 07      LDAB 7,X
FA00 8D C0      BSR  ASCLFT
FA02 E7 06      STAB 6,X
FA04 E6 07      LDAB 7,X
FA06 8D BE      BSR  ASCRT
FA08 E7 07      STAB 7,X
FA0A 20 0A      BRA  SETCR
               *
               * ALL READS USE AN EXTRA
               * BLOCK CT OF 255.
               *
FA0C C6 32     MAXCT  LDAB #'2
FA0E E7 05      STAB 5,X
FA10 C6 35      LDAB #'5
FA12 E7 06      STAB 6,X
FA14 E7 07      STAB 7,X
FA16 C6 0D     SETCR  LDAB #CR
```



```
FA18 E7 00      STAB 0,X
FA1A 39         RTS
               *
               * ALL READS USE A MAX
               * BLOCK CT AS WE DON'T
               * KNOW THE BLK CT.  THE
               * 1ST 2 BYTES READ ARE
               * THE BYTE CT WHICH IS
               * USED TO SET UP 'ENDADR'
               * AT $3E  WHEN THE INDEX
               * MATCHES 'ENDADR', READ
               * INTO MEM STOPS AND IF
               * THE B REG = 0 WE'RE
               * DONE.  ELSE, THE BETA
               * READ DATA IS ACCEPTED
               * TILL THAT BLK IS DONE
               *
               * NOTE: BETA XMITS $07
               * TWICE IF NORMAL DATA!!
               *
FA1B C6 12     READ  LDAB #RDCMD
FA1D BD F97C    JSR  SETUP
FA20 BD F930    JSR  RDBETA  RD LO BYTCT
FA23 81 07      CMPA #BELL
FA25 26 07      BNE  SAVLO
FA27 BD F930    JSR  RDBETA
FA2A 81 07      CMPA #BELL
FA2C 26 20      BNE  CTERR
FA2E 16        SAVLO  TAB
FA2F BD F930    JSR  RDBETA  RD HI BYTCT
FA32 81 07      CMPA #BELL
FA34 26 07      BNE  FIXEND
FA36 BD F930    JSR  RDBETA
FA39 81 07      CMPA #BELL
FA3B 26 10      BNE  CTERR
FA3D DD 3D     FIXEND ADDB BEGADR+1
FA3F 99 3C      ADCA BEGADR
FA41 D7 3F      STAB ENDADR+1
FA43 97 3E      STAA ENDADR
FA45 C6 FE      LDAB #256-2
FA47 DE 3C      LDX  BEGADR
FA49 BD F930   RD1  JSR  RDBETA
FA4C A7 00      STAA 0,X
FA4E 81 07      CMPA #BELL
FA50 26 0B      BNE  RD2
FA52 BD F930    JSR  RDBETA
FA55 81 07      CMPA #BELL
FA57 27 04      BEQ  RD2
               *
FA59           RDERR EQU *
FA59 5F        CTERR  CLRB
FA5A 7E F8E3    JMP  HAVERR
               *
FA5D 5A        RD2  DECB
```

```
FA5E 9C 3E      CPX  ENDADR
FA60 27 03      BEQ  CKRPAD
FA62 08         INX
FA63 20 E4      BRA  RD1
FA65 5D        CKRPAD TSTB
FA66 27 0A      BEQ  NORPAD
FA68 BD F930   RDPAD  JSR  RDBETA
FA6B 81 07      CMPA #BELL
FA6D 27 EA      BEQ  RDERR
FA6F 5A         DECB
FA70 26 F6      BNE  RDPAD
                *
                * THIS DELAY ALLOWS BETA
                * TO SEND ERROR INFO IF
                * IT OCCURRED ON END BLK
                *
FA72 C6 41     NORPAD LDAB #65
FA74 5A        DLY  DECB
FA75 26 FD      BNE  DLY
FA77 7D F041    TST  BRDCTL
FA7A 2A 03      BPL  NRDERR
FA7C 7E F806    JMP  ENDOP
                *
                * NOW WE TELL THE BETA TO
                * ABORT THE REMAINDER OF
                * THE 256 BLOCK READ BY
                * AN ESCAPE SEQUENCE.
                *
FA7F 86 1B     NRDERR LDAA #ESC
FA81 BD F922    JSR  WRBETA
FA84 86 0D      LDAA #CR
FA86 BD F922    JSR  WRBETA
FA89 7E F806    JMP  ENDOP
                *
FA8C C6 16     VERIFY LDAB #VERCMD
FA8E BD F97C    JSR  SETUP
FA91 7E F806    JMP  ENDOP
                *
                * THIS CODE IS THE MECA
                * SUPPLIED FLOWCHART FOR
                * INITIAL BRINGUP OF THE
                * BETA DRIVE.....
                *
FA94 BD F86A   BETMON JSR  INIT
FA97 8D 18     LOOP   BSR  TOBETA
FA99 8D 02      BSR  TOCONS
FA9B 20 FA      BRA  LOOP
                *
FA9D B6 F041   TOCONS LDAA BRDCTL
FAA0 2A 0E      BPL  RTN2
FAA2 B6 F040    LDAA BRDATA
FAA5 81 07      CMPA #BELL
FAA7 27 07      BEQ  RTN2
FAA9 81 0A      CMPA #LF
```



```
FAAB 27 03      BEQ  RTN2
FAAD 7E FCAD    JMP  PUTCHR
FAB0 39        RTN2 RTS
                *
FAB1 B6 F043   TOBETA LDAA BWRCTL
FAB4 2A FA      BPL  RTN2
FAB6 B6 F001    LDAA KBDREQ
FAB9 2A F5      BPL  RTN2
FABB B6 F000    LDAA KBDATA
FABE F6 F042    LDAB BWRDAT
FAC1 81 22      CMPA #$22
FAC3 26 05      BNE  WRB
FAC5 31         INS
FAC6 31         INS
FAC7 7E FE4F    JMP  DEBUG
                *
FACA B7 F042   WRB STAA BWRDAT
FACD 39         RTS
                OPT  PAG,PRT,NOG
                END
```

NO ERROR(S) DETECTED

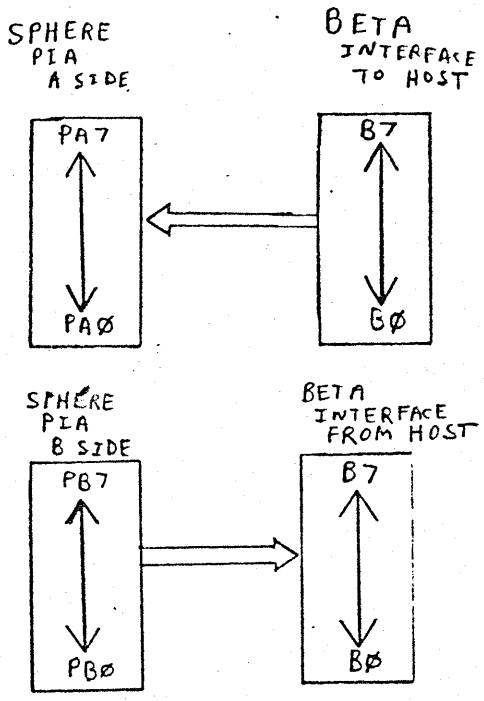
SYMBOL TABLE:

AAPGM	F826	ASCLFT	F9C2
ASCRT	F9C6	BEGADR	003C
BELL	0007	BETMON	FA94
BINIT	F840	BINREW	F858
BLKNO	0033	BRDATA	F040
BRDCTL	F041	BREAD	F84B
BWRCTL	F043	BWRDAT	F042
BWRITE	F846	CKBELL	F8D0
CKRPAD	FA65	CKSWI	F829
CKWPAD	F976	CLRCMD	0010
CR	000D	CT2ASC	F9F8
CT2DEC	F9E7	CTDOWN	F898
CTERR	FA59	DEBG	F843
DEBUG	FE4F	DLY	FA74
DOENQ	F8F7	DSPRD	F888
DSPWR	F86E	ENDADR	003E
ENDOP	F806	ENQ	F903
ENQCMD	0500	ERRE	F8BB
ERRIND	E0BF	ERRN	F8BF
ERROR	F912	ESC	001B
FIXEND	FA3D	HAVERR	F8E3
INIT	F86A	INTREW	F866
KBDATA	F000	KBDREQ	F001
LF	000A	LINE2	E040
LOOP	FA97	MAXCT	FA0C
NOPRNT	003A	NORPAD	FA72
NOWPAD	F979	NRDERR	FA7F
NXTBNT	F8C7	PUTCHR	FCAD
RD1	FA49	RD2	FA50

```
ADBETA F936    RDCMD  0012
RDERR  FA59    RDLF   F8E6
RDPRO  FA68    RERD   FA1B
RERD2  F85C    REWCMD 1A80
REWIND F862    RTN1   F8AA
RTN2   FA80    RWIND  F88E
SAVLO  FA2E    SELTRK F989
SETBLK F9A8    SETCR  FA16
SETCT  F9CB    SETUP  F97C
SNDCMD F8A8    SNDRCV F919
STATUS 0009    SWIFLG BFF9
TOBETA FA81    TOCONS FA90
TRKCMD 0014    UIRVEC BFFA
UNMVEC BFF0    USWI   F831
USWVEC BFF6    VERCMD 0016
VERIF2 F85F    VERIFY FA8C
WCT2   F953    WCTERR F950
WKCMD  0004    WR1    F95D
WR2    F968    WRD    FACA
WRBETA F922    WRCMD  0017
WRITE  F938    WRITE2 F850
WRPRO  F973    WTELF  F89A
WTELF2 F9A0

SAVE SOURCE?
```

SPHERE ⟺ BETA INTERFACE



CA2 ⟶ $\overline{\text{DATA ACK}}$ * 470 Ω terminating resistor in Beta was removed

CA1 ⟵ $\overline{\text{OBF}}$ (minus when output buffer is full)

CB2 ⟶ $\overline{\text{INPUT STROBE}}$ * 470 Ω terminating resistor in Beta was removed

CB1 ⟵ DATA RCVD

JPC TO SPHERE

INTRODUCTION -

THE TC-3 CASSETTE INTERFACE BOARD PROVIDES HIGH PERFORMANCE PROGRAM STORAGE FOR SWTPC COMPUTER SYSTEMS. THE INTERFACE BOARD PLUGS INTO ONE I/O SLOT OF THE SWTPC MOTHERBOARD. ELIMINATING THE INCONVENIENCE OF A SEPARATE CABINET. CONNECTION IS MADE TO A STANDARD CASSETTE RECORDER THROUGH TWO AUDIO CABLES.

THE MAJOR DESIGN GOALS FOR THE TC-3 WERE SPEED AND DATA RELIABILITY. THE INTERFACE OPERATES AT 4800 BAUD AND LOADS A 4K FILE IN 8 SECONDS. DATA IS RECORDED IN A MODIFIED FM FORMAT SIMILAR TO DISK SYSTEMS, AND THE SPEED AND RELIABILITY OF THE BOARD ARE COMPARABLE TO THOSE OF MINI-FLOPPY DISKS.

IN ADDITION TO READING AND WRITING IN HIGH SPEED BINARY, THE TC-3 IS CAPABLE OF READING OR WRITING CASSETTES IN KANSAS CITY STANDARD FORMAT WITHOUT ANY MODIFICATION TO THE INTERFACE BOARD; ALL FORMATTING IS UNDER SOFTWARE CONTROL.

THE TC-3 ALSO PROVIDES A FULLY BUFFERED 8-BIT OUTPUT PORT CAPABLE OF DIRECTLY SINKING 40 MA AT 30 VOLTS. THE PORT HAS FULL HANDSHAKE AND INTERRUPT CAPABILITY FOR USE AS A PARALLEL DATA OUTPUT PORT, OR CAN BE USED AS DISCRETE OUTPUT LINES TO CONTROL THE CASSETTE TRANSPORT.

CASSETTE RECORDERS AND TAPES -

THE TC-3 IS A VERY HIGH PERFORMANCE DEVICE ... IT REQUIRES A DECENT RECORDER AND QUALITY TAPES TO BE ABLE TO DELIVER THE PERFORMANCE OF WHICH IT IS CAPABLE. THE IDEAL RECORDER FOR USE WITH THE TC-3 IS A BOTTOM-OF-THE-LINE STEREO CASSETTE DECK. A DECK DIFFERS FROM A RECORDER IN THAT IT IS DESIGNED TO BE A COMPONENT IN AN AUDIO SYSTEM AND THEREFORE HAS NO BUILT-IN SPEAKERS, OR AMPLIFIERS. THE PREAMPLIFIER SECTION IS USUALLY DESIGNED TO PROVIDE MUCH BETTER FREQUENCY RESPONSE THAN SMALL RECORDERS WITH THEIR BUILT-IN AMPLIFIERS. CASSETTE DECKS HAVE ADDITIONAL FEATURES THAT MAKE THEM DESIRABLE IN THIS APPLICATION....THEY GENERALLY HAVE VU METERS, ARE BUILT BETTER, HAVE GOOD FOOTAGE COUNTERS, HAVE SHORTER REWIND TIMES, COME WITH THE REQUIRED AUDIO CABLES, AND THE REDUNDANCY OF STEREO RECORDING IMPROVES THE DATA RELIABILITY. WE HAVE TESTED SEVERAL DECKS RANGING IN PRICE FROM $80 TO $150 AND ALL OF THEM PERFORMED FLAWLESSLY AT 4800 BAUD. THERE ARE SOME RECORDERS IN THE $60 CLASS THAT WORK WELL AT 2400 BAUD, BUT ARE NOT GENERALLY RELIABLE AT 4800 BAUD, AND THEY HAVE OTHER DISADVANTAGES AS WELL.

CASSETTE TAPES ARE CRITICALLY IMPORTANT TO GOOD RESULTS. IT IS NOT WORTHWHILE EXPERIMENTING WITH CHEAP TAPE...IT JUST WON'T PROVIDE RELIABLE PERFORMANCE. WE HAVE CONSISTENTLY HAD THE BEST RESULTS WITH MEMOREX MRX-3 IN 30 OR 60 MINUTE LENGTH. OTHER HIGH OUTPUT LOW NOISE CASSETTES MAY ALSO WORK WELL. DO NOT USE 90 MINUTE CASSETTES; THEY ARE THINNER AND TEND TO JAM, AND HAVE MORE PROBLEMS WITH DROPOUT AND PRINT-THROUGH.

CONNECT THE JPC TC-3 HIGH SPEED CASSETTE SYSTEM TO THE SPHERE - 2400 BAUD TO LOAD 8K BASIC IN 45 SECONDS RELIABILITY -

Chan Wai Yung
P.O.BOX K-2296
Kowloon, Hongkong

The TC-3 is used with 600, 1200, 2400, 4800 and 9600 baud by software to control. After my testing it runs very reliable with 2400 baud im my desk recorder. The frequency is like FM format, so it is difficult to operate in portable recorder. Desk recorder has a good response to the audio frequency.

HARDWARE PART: I built a broad and address F090 for the TC-3 PIA I/O port and connect the TC-3 as figure.

SOFTWARE PART: The TC-3 software program is relocatable code, can be enter wherever available memory. I modify as:

0004 B7 8011 to B7 F091	0016 7F A04F to 7F 002A	0019 FE A002 to FE 003c
0022 BB A04F to BB 002A	0025 B7 A04F to B7 002A	0028 BC A004 to BC 003E
002F BI A04F to BI 002A	0034 7E E0E3 to 39 0101	003A F7 A04D to F7 0028
0045 7A A04d to 7A 0028	005D F6 8010 to F6 F090	0062 FI A04E to FI 0029
0067 F7 A04E to F7 0029	006E 7F 8011 to 7F F09I	0073 B7 90I0 to B7 F090
0078 B7 8011 to B7 F09I	0087 7F A04F to 7F 002A	008A FE A002 to FE 003C
0095 BB A04F to BB 002A	0098 B7 A04F to B7 002A	009B BC A004 to BC 003E
00A0 b6 A04F to B6 002A	00A5 7E E0E3 to 39 0101	00AE F7 A04D to F7 0028
00C1 7A A04D to 7A 0028	00CF F6 80I0 to F6 F090	00DA F6 80I0 to F6 F090
00DF F7 80I0 to F7 F090	00E4 F6 80I0 to F6 F090	00E9 F7 80I0 to F7 F090

SAVE DATA TO TAPE:

1. Load TC-3 software program in Sphere memory.
2. Open 003c to XXXX begin address of data.
3. Open 003E to YYYY end addres of data.
4. Open write addres of TC-3 software.
5. Desk recorder in record mode for a moment.
6. Control J

LOAD DATA FROM TAPE:

I. Load TC-3 software program in Sphere memory.
2. Open 003C to XXXX begin address.
3. Open 003E to YYYY end address.
4. Open read address of TC-3 software.
5. Desk recorder in play mode.
6. Control J

I have never one bad load with 2400 baud.....

CONNECT THE JPC TC-3 HIGH SPEED CASSETTE SYSTEM TO THE SPHERE - 2400 BAUD TO LOAD 8K BASIC IN 45 SECONDS RELIABILITY -

Chan Wai Yung
P.O.BOX K-2296
Kowloon, Hongkong

The TC-3 is used with 600, 1200, 2400, 4800 and 9600 baud by software to control. After my testing it runs very reliable with 2400 baud in my desk recorder. The frequency is like FM format, so it is difficult to operate in portable recorder. Desk recorder has a good response to the audio frequency.

HARDWARE PART: I built a broad and address F090 for the TC-3 PIA I/O port and connect the TC-3 as figure.

SOFTWARE PART: The TC-3 software program is relocatable code, can be enter wherever available memory. I modify as:

0004 B7 8011 to B7 F091	0016 7F A04F to 7F 002A	0019 FE A002 to FE 003c
0022 BB A04F to BB 002A	0025 B7 A04F to B7 002A	0028 BC A004 to BC 003E
002F BI A04F to BI 002A	0034 7E E0E3 to 39 0101	003A F7 A04D to F7 0028
0045 7A A04d to 7A 0028	005D F6 8010 to F6 F090	0062 FI A04E to FI 0029
0067 F7 A04E to F7 0029	006E 7F 8011 to 7F F09I	0073 B7 90I0 to B7 F090
0078 B7 9011 to B7 F09I	0087 7F A04F to 7F 002A	008A FE A002 to FE 003C
0095 BB A04F to BB 002A	0098 B7 A04F to B7 002A	009B BC A004 to BC 003E
00A0 b6 A04F to B6 002A	00A5 7E E0E3 to 39 0101	00AE F7 A04D to F7 0028
00CI 7A A04D to 7A 0028	00CF F6 80I0 to F6 F090	00DA F6 80I0 to F6 F090
00DF F7 80I0 to F7 F090	00E4 F6 80I0 to F6 F090	00E9 F7 80I0 to F7 F090

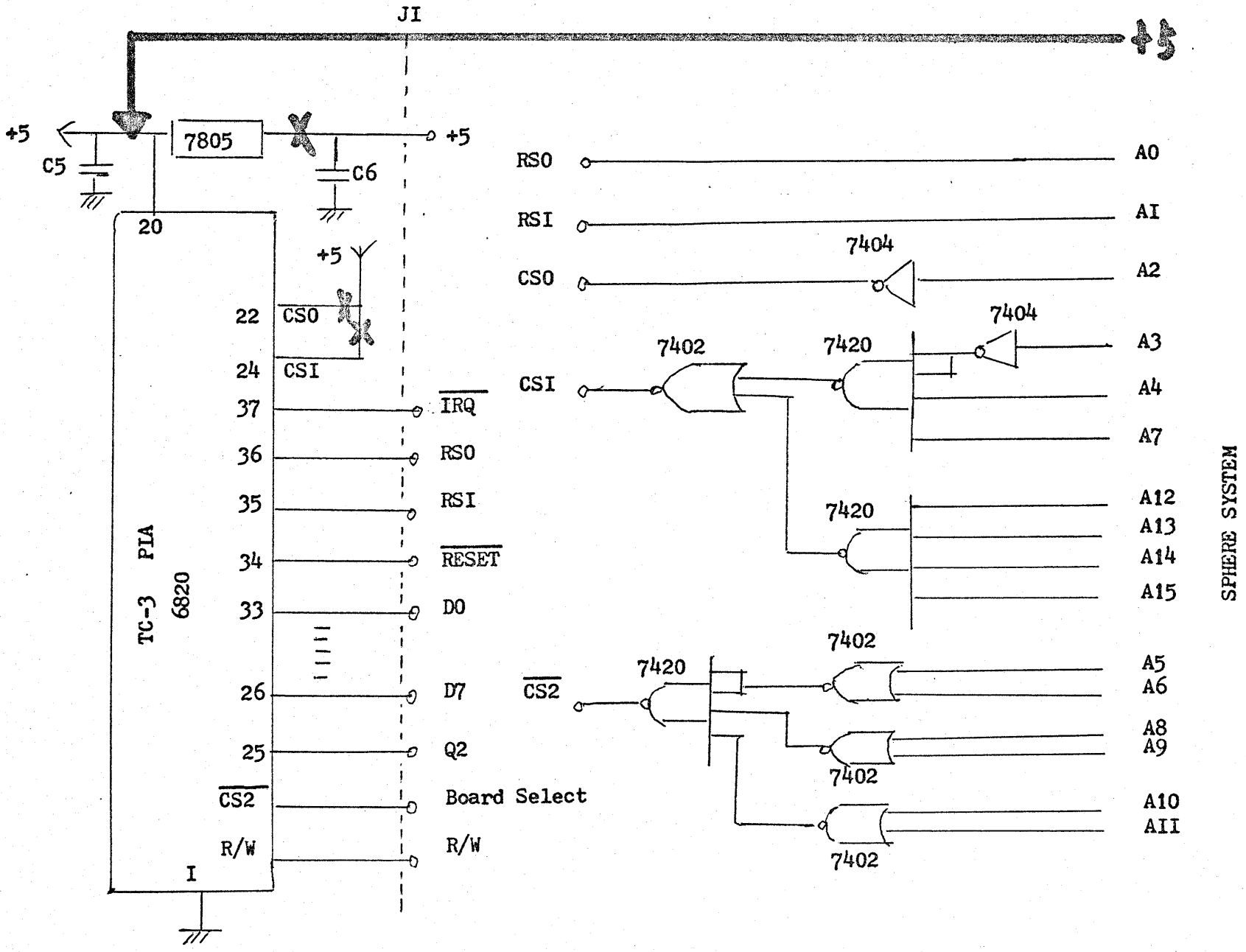
SAVE DATA TO TAPE:

1. Load TC-3 software program in Sphere memory.
2. Open 003c to XXXX begin address of data.
3. Open 003E to YYYY end addres of data.
4. Open write addres of TC-3 software.
5. Desk recorder in record mode for a moment.
6. Control J

LOAD DATA FROM TAPE:

I. Load TC-3 software program in Sphere memory.
2. Open 003C to XXXX begin address.
3. Open 003E to YYYY end address.
4. Open read address of TC-3 software.
5. Desk recorder in play mode.
6. Control J

I have never one bad load with 2400 baud.....


JI
+5
7805
+5
C5
C6
20
+5
22
CS0
24
CSI
37
IRQ
36
RS0
35
RSI
34
RESET
33
D0
26
D7
25
Q2
CS2
Board Select
R/W
R/W
I
TC-3 PIA
6820
RS0
RSI
CS0
CSI
CS2
7404
7402
7420
7404
7402
7420
7420
7402
7402
A0
AI
A2
A3
A4
A7
A12
A13
A14
A15
A5
A6
A8
A9
A10
AII
SPHERE SYSTEM

CSS BASIC with JPC TC-3 high speed save and load

```
4C 4F 41 44 00 2D 00 53 41 56 45 00 2D 46 54 41 50 00 2D 6C
54 43 48 00 2D 77
```

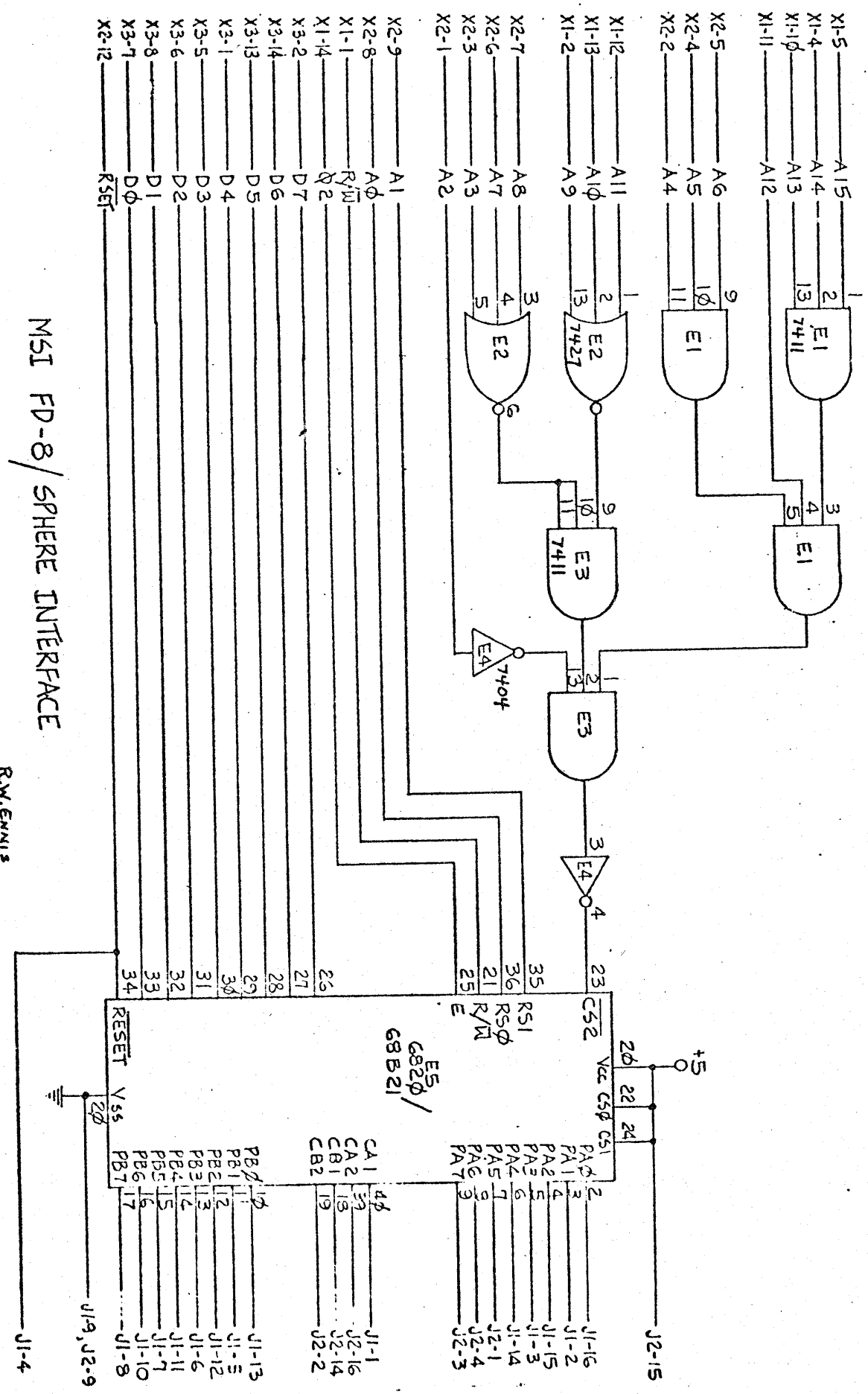
SAVE	save BASIC program to tape, as long as finish, a subroutine will create 4 digital hexadecimal code to appear - size of program length-. This 4 code is as a block name of this BASIC program. Example 0IA4 appear, now IA4 is this program to be name.
LOAD XXXX	load BASIC program from tape. Example load 1A4 (return Key) the name IA4 BASIC program to be load.
TAP XXXX	append BAIC program from tape. as load command.
TCH XXXX	chain BASIC program from tape.

```
2D00  BD 0A 54 DE 3C DF EC DE 3E DF EE DE 20 DF 3C DE 22 DF 3E
      DE 2C BD 09 63 8D 06 D7 3E 97 3F 20 05 BD FF 22 08 39 96
      3D 9B 3F 97 3F 96 3C 99 3E 97 3E BD XX XX DE 3E DF 22 DF
      24 CE 99 99 DF 3C DF 3E 39 20 BC 20 BD DE 20 DF 3C DE 22
      DF 3E BD YY YY 96 3E D6 3F D0 3D 92 3C 97 3E D7 3F 86 57
      97 ID DE 3E 8D 02 20 D3 DF 40 7E FE FC DE 20 DF IE DE 22
      DF 20 8D CE DE IE DF 20 39 8D C5 7E 0A C6
```

XXXX = TC-3 routine Read adress.
YYYY = TC-3 write addres.

Editor's Note:
We cannot legally print the full JPC software or schematics. The price is reasonable for 2400 baud cassette capabilities. Also, your SIM board could be used for other serial peripheral devices instead.

PART OF THE OPERATING SOFTWARE IS IN DECEMBER 1979 ISSUE. WRITE TO BOE ENNIS, 9322 LAUREL AVE., FONTANA, CA. 92335 for help and the rest of the SOFTWARE.



```
FIND  DE LDX  2C
      BD JSR  0609
      DF STX  2C
      DE LDX  63
      39 RTS
TART  8D BSR  F4      FIND
      DF STX  6E
      09 DEX
      09 DEX
      09 DEX
      09 DEX
      EE LDX X00
      DF STX  6A
      8D BSR  E8      FIND
      DF STX  A2
      09 DEX
      09 DEX
      09 DEX
      09 DEX
      EE LDX X00
      DF STX  6C
      CE LDX #006A                TEST DIMS
      A6 LDAAX00
      A1 CMPAX03
      26 BNE  06      ERR
      A6 LDAAX01
      A1 CMPAX02
      27 BEQ  03      DIMSOK
ERR   7E JMP  132F                DIM ERROR
DIMSOK 6D TST X32
      27 BEQ  06      BAS1        FIX FOR BASE=0
      6C INC X00
      6C INC X01
      6C INC X02
BAS1  D6 LDAB 6A
LOOP  37 PSHB                     GET VARIABLE POINTER
      DE LDX  6E                  GET VALUE
      BD JSR  0353                GET DESTINATION MATRIX POINTER
      DE LDX  A2                  STORE TRANSPOSED VALUE
      BD JSR  031E
      DE LDX  6E
      8D BSR  20      INX6
      DF STX  6E
      DE LDX  A2
      96 LDAA 6B
INX   8D BSR  18      INX6
      4A DECA
      26 BNE  FB      INX
      DF STX  A2
      33 PULB
      5A DECB
      26 BNE  E0      LOOP
      D6 LDAB 6C
      DE LDX  63
      8D BSR  09      INX6
      DF STX  A2
      DF STX  63
      7A DEC  006C
      26 BNE  CF      BAS1
INX6  08 INX
      08 INX
      08 INX
      08 INX
      08 INX
      08 INX                      EXIT
```

SAMPLE PROGRAM

```
0010 RJUST= 3
0020 DIM A(3,3),B(3,3)
0030 MATREAD A(1,1)
0040 MATTRANS A(1,1) B(1,1)
0050 MATPRINT A(1,1);PRINT
0060 MATPRINT B(1,1)
0070 DATA 1,2,3,4,5,6,7,8,9
```

```
 1   4   7
 2   5   8
 3   6   9

 1   2   3
 4   5   6
 7   8   9
```

MAT IDN

```
       DE LDX  2C
       BD JSR  0609
       DF STX  2C
       DE LDX  63
       09 DEX
       09 DEX
       09 DEX
       09 DEX
       E6 LDABX00
       08 INX
       A6 LDAAX00
       11 CBA
       27 BEQ  03        DIMSOK
       7E JMP  132F
DIMSOK 97 STAA 6D
       08 INX
       A6 LDAAX00
       97 STAA 6E
       08 INX
       A6 LDAAX00
       97 STAA 6F
       08 INX
CLEAR  6F CLR X00
       08 INX
       9C CPX  6E
       26 BNE  F9        CLEAR
       09 DEX
PUT1   86 LDAA#01
       A7 STAAX00
       BD JSR  04BC
       A7 STAAX00
       9C CPX  63
       26 BNE  01        CONT
       39 RTS
       09 DEX
CONT   D6 LDAB 6D
       7D TST  009C
       27 BEQ  01        DECX
       5C INCB
DECX   BD JSR  04BB
       5A DECB
       26 BNE  FA        DECX
       20 BRA  E1        PUT1
```

ERROR MSG. DIM ERROR

EXIT

SAMPLE PROGRAM

```
0010 DIM A(6,6)
0020 MATIDN A(1,1)
0030 MATPRINT A(1,1)

 1  0  0  0  0  0
 0  1  0  0  0  0
 0  0  1  0  0  0
 0  0  0  1  0  0
 0  0  0  0  1  0
 0  0  0  0  0  1
```

SUNICK
SYSTEMS

185 SUMMIT DRIVE
SANTA CRUZ, CA.
95060

December 8, 1975

Sphere Corp.

791 South 500 West

Bountiful , Utah 84010

Dear Eric,

I appreciate your returning my phone call this afternoon. I find your company very hard, if not impossible to communicate with by mail. I guess return mail is on the rock bottom of your priority list.

I was very interested by your comments on my 2-VMA method of refresh. When I originally designed the modification, I knew much less about the system than I do now. I put the modification in and the system worked for all the simple programs that I was running. I had been noticing lately that every once in awhile, the system would blow up when I would try to re-edit an assembler program.

Today I sat down and wrote a very simple program that locks the system really tight. No refresh occurs at all, except in the tight loop that the program is in.

```
700  86
701  00
702  7E
703  07
704  00
```

Included on a seperate sheet is the very simple fix for this problem, together with the entire mod in case you filed the previous one.

I have tried it with the above tight loop and it works perfectly. I hope I caught you before you sent the next news letter out. Also included are some questions that you might answer if you get time.

Regards,

William Nichparenko

SUNICK
SYSTEMS

185 SUMMIT DRIVE
SANTA CRUZ, CA.
95060

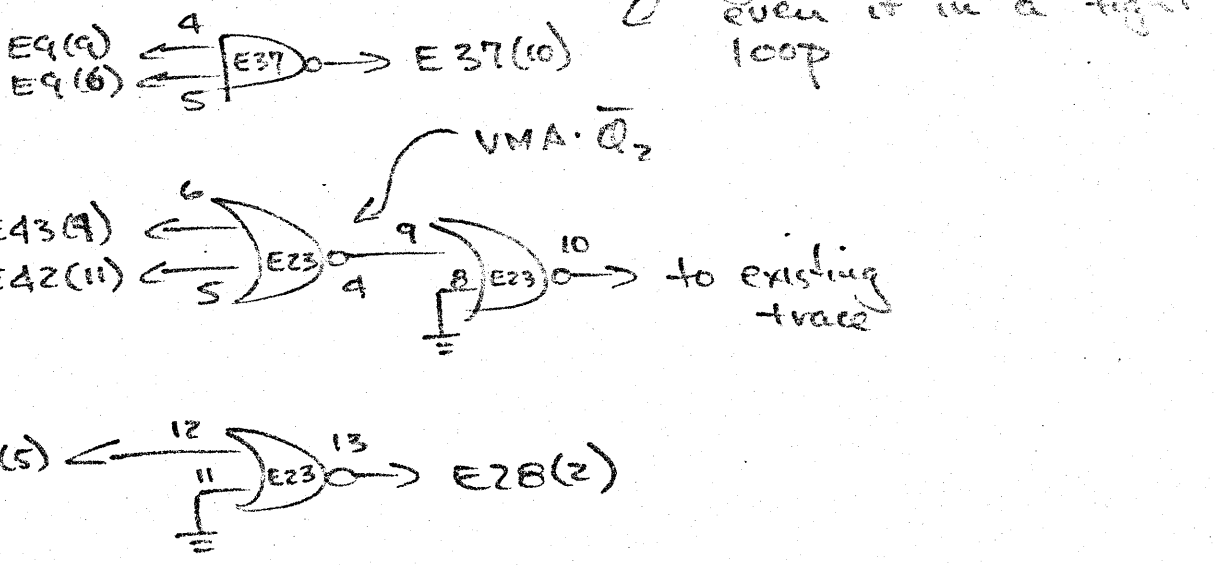
December 8, 1975

& 2 VMA Method of Invisible Refresh.

1) Cut the following lines: E37(10) to E22(3). E35(9) to E23(8). E47(7) to E23(9). This isolates E23(8,9) and E37(10).

2) Cut E28(2) to E28(13), E28(2) to E45(1) then jumper E28(13) to E45(1). Cut E28(12) to E9(3), E28(12) to E35(3) then jumper E9(3) to E35(3). Both E28(2) and E28(12) ane tie points and need to be isolated. A jumper must then be run to re-connect the old trace.

3) Do the following wiring:



William Nichparenko

Sunick Systems

February 5, 1976

Mr. Eric Jameson
Sphere Corp.
791 South 500 West
Bountiful,Utah 84010

Dear Eric,

In reference to the LRC printers, they are $195, 1-9,$186, 10-49,$167, 50-99.

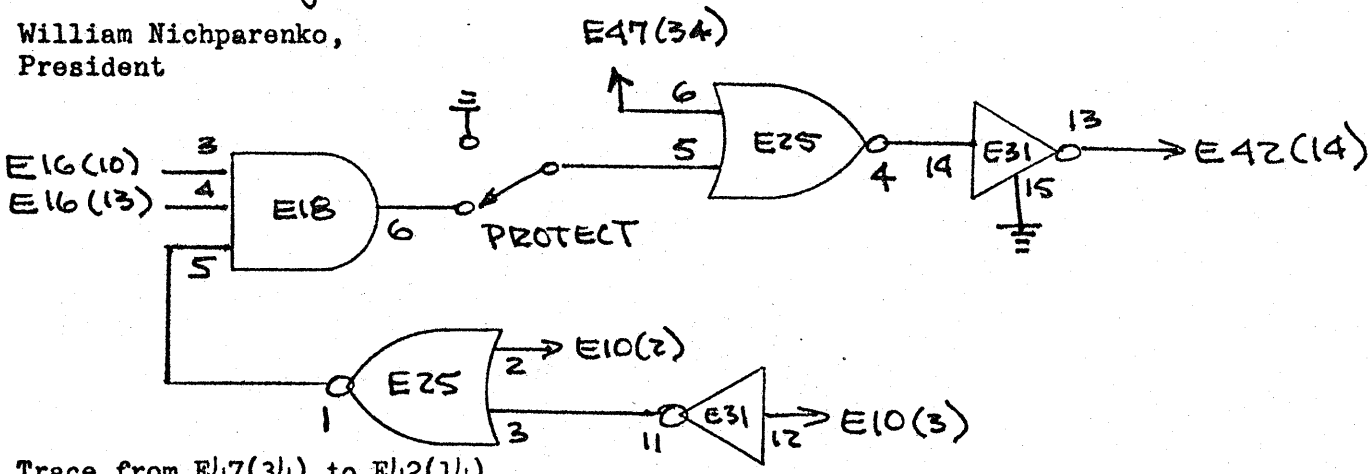
Below is a diagram for the memory protect circuit. It needs only the added 7402 in the blank socket on the CPU board.

I would very much appreciate a copy of BASIC as soon as it comes out. Hopefully I can read it on my tape system, since that is what the Kansas City standard is all about. The only information I need to know is how you pack the data in the eight bit field, and what sort of lables you use. I can then adjust my software accordingly. A short description of the commands would also be helpful.

Granted your newsletter will be out in 2 to 3 weeks, but if my experience with the last newsletter holds true, the projected date will probably be delayed some, and it takes two weeks to get here by pony express.

Regards,

William Nichparenko,
President



Trace from E47(34) to E42(14) must be cut.
Trace from E42(15) to E42(1) and E30(6) to E42(15) must be cut. Then connect E42(1) to E30(6), E42(15) to Ground.
The above defeats the write pulse during any addresses 800 to BFF hex. This allows normal editing of assembler programs around the protected program. If desired a protected assembler programs may be stored in the protected space, but 08 must first be loaded with 800, then re-edit via FC68. Location 20 must be loaded with the loaction of the last assembler byte also.

185 Summit Drive Santa Cruz, CA 95060 408:426-5336

```
0010 REM INVOICING PROGRAM CSS/SWTP BASIC FOR SYSTEM.
0020 DIM V(10), X(10)
0030 DIM A$(6), B$(10)
0040 DIM Z(10), W(10)
0050 PRINT\"MAKING INVOICE PROGRAM"
0060 PRINT
0070 DIGITS= 0
0080 PRINT\"SELL TO COMPANY NAME"
0090 LINPUT A$(1)
0100 PRINT\"ADDRESS"
0110 LINPUT A$(2)
0120 PRINT\"ZONE"
0130 LINPUT A$(3)
0140 PRINT\"INOIVE DATE"
0150 INPUT A$(4)
0160 PRINT\"INVOICE (NO)"
0170 INPUT A$(5)
0180 PRINT\"HOW ITEMS"
0190 INPUT M
0200 FOR K=1 TO M
0210 PRINT\"ITEM "K
0220 PRINT\"MODEL (NO)"
0230 INPUT W(K)
0240 PRINT\"QUANTITY"
0250 INPUT Z(K)
0260 PRINT\"BRAND NAME"
0270 INPUT B$(K)
0280 PRINT\"UNIT PRICE"
0290 INPUT V(K)
0300 LET X(K)=Z(K)(AS)V(K)
0310 NEXT K
0320 PRINT\"HAVE OR NOT SPARE PARTS"
0330 INPUT C$
0340 PRINT\"DISCOUNT %"
0350 INPUT D
0360 FOR K=1TOM
0370 LET X(K)=INT((X(K)(AS)1000+5)/10)/100
0380 LET X=X+X(K)
0390 NEXT K
0400 LET X=INT((X(AS)1000+5)/10)/100
0410 LET X9=X
0420 PRINT :PRINT
0430 PRINT\"PRINT OUT FROM 1=CRT, 7=TTY, 9=READY"
0440 INPUT G
0450 LINE= 0
0460 IF G=9 THEN END
0470 PRINT (NO)G
0480 PRINT (NO)G,TAB(4) A$(1)
0490 PRINT (NO)G,TAB(4) A$(2)
0500 PRINT (NO)G,TAB(4) A$(3)
```

```
0510 PRINT (NO)G,TAB(52);A$(4)
0520 PRINT (NO)G:PRINT(NO)G
0530 PRINT (NO)G,TAB(26);"INVOICE NO. " A$(5)
0540 PRINT (NO)G,TAB(26);"----------------"
0550 PRINT (NO)G:PRINT(NO)G
0560 FOR K=1TOM
0570 PRINT (NO)G,TAB(2);Z(K);"SETS ";TAB(12);"MODEL ";W(K);"'";
     B$(K);"'"
0580 DIGITS= 2
0590 LET T=INT(LOG(X(K))/LOG(10))
0600 IF X(K)(GT)=1000 GOTO 1490
0610 LET T=INT(LOG(X(K))/LOG(10))
0620 PRINT (NO)G," BRAND RADIO." TAB(48);"@US";V(K);TAB(64-T);"
     US";X(K)
0630 IF W(K)=710 GOSUB 710
0640 IF W(K)=503 GOSUB 760
0650 IF W(K)=502 GOSUB 760
0660 IF W(K)=3302 GOSUB 810
0670 DIGITS= 0
0680 PRINT (NO)G
0690 NEXT K
0700 GOTO 890
0710 PRINT (NO)G,TAB(12) "8 SOLID STATE PORTABLE RADIO.(6 TR."
0720 PRINT (NO)G,TAB(12) "2 DIODE) FULL CIRCUIT, SW/MW BAND."
0730 PRINT (NO)G,TAB(12) "COMPLETED WITH EARPHONE, UM-1 X 3"
0740 PRINT (NO)G,TAB(12) "BATTERIES, IN PRINTED GIFT BOX."
0750 RETURN
0760 PRINT (NO)G,TAB(12) "6 SOLID STATE POCKET RADIO.(5 TR."
0770 PRINT (NO)G,TAB(12) "2 DIODE) 2-IFT, MW BAND, COMPLETE"
0780 PRINT (NO)G,TAB(12) "WITH UM-3 X 2 BATTERIES, IN"
0790 PRINT (NO)G,TAB(12) "PRINTED GIFT BOX."
0800 RETURN
0810 PRINT (NO)G,TAB(12) "10 SOLID STATE PORTABLE RADIO.(8 TR."
0820 PRINT (NO)G,TAB(12) "2 DIODE) FULL CIRCUIT, SW/MW BAND."
0830 PRINT (NO)G,TAB(12) "COMPLETED WITH UM-1 X 2 BATTERIES,"
0840 PRINT (NO)G,TAB(12) "EARPHONE, CARRYING CASE, IN PRINTED"
0850 PRINT (NO)G,TAB(12) "GIFT BOX.":RETURN
0860 LET T=INT(LOG(X)/LOG(10)):IF X(GT)=1000 GOTO 1530
0870 PRINT (NO)G,TAB(48) "TOTAL: ";TAB(64-T);"US";X:RETURN
0880 PRINT (NO)G,TAB(12) "1% SPARE PARTS FREE OF CHARGE.":RETUR
     N
0890 DIGITS= 2:GOSUB 860:IF C$="HAVE" GOSUB 880
0900 IF D=0 GOTO 1020
0910 DIGITS= 0
0920 LET D$=STR$(D)
0930 LET D1=X(AS)(D/100):D1=INT((D1(AS)1000+5)/10)/100
0940 PRINT (NO)G,TAB(12);D$;"% TO BE LESS DISCOUNT: ";
0950 LET T=INT(LOG(D1)/LOG(10)):DIGITS=2
0960 IF T(LT)0 THEN T=0
0970 IF D1(GT)=1000 GOTO 1570
0980 PRINT (NO)G,TAB(64-T);"US" D1
0990 LET X=X-D1
```

```
1000 LET T=INT(LOG(X)/LOG(10)):IF X(GT)=1000 GOTO 1610
1010 PRINT (NO)G,TAB(48);"TOTAL:"; TAB(64-T);"US"; X
1020 PRINT (NO)G
1030 PRINT (NO)G,TAB(12)"TOTAL U.S. DOLLARS ";
1040 IF X(GT)=100000 GOTO 1060
1050 GOTO 1140
1060 LET Y=100000
1070 GOSUB 1650
1080 GOSUB 2060
1090 PRINT (NO)G,"HUNDERED ";
1100 LET X=(X1-X2)(AS)Y
1110 GOSUB 2060
1120 IF X(LT)999.99 THEN PRINT(NO)G,"THOUSAND ";
1130 IF X=0 THEN PRINT(NO)G,"THOUSAND ";
1140 IF X(GT)=1000 GOTO 1160
1150 GOTO 1230
1160 LET Y=1000
1170 GOSUB 2060
1180 GOSUB 1650
1190 GOSUB 2060
1200 IF (X1-(X2(AS)10))(AS)1000(GT)=1000 THEN X=(X1-(X2(AS)1
     0))(AS)1000:GOTO 1160
1210 PRINT (NO)G,"THOUSAND ";
1220 LET X=(X1-X2)(AS)Y
1230 IF X(GT)=100 GOTO 1250
1240 GOTO 1290
1250 LET Y=100
1260 GOSUB 2060
1270 GOSUB 1650:PRINT(NO)G,"HUNDRED ";
1280 LET X=(X1-X2)(AS)Y
1290 IF X(GT)=1 GOTO 1310
1300 GOTO 1360
1310 LET Y=1
1320 GOSUB 2060
1330 GOSUB 1650
1340 IF X1-(X2(AS)10)(GT)=1 THEN X=X1-(X2(AS)10):GOTO 1310
1350 LET X=(X1-X2)(AS)Y
1360 IF X(GT)=.01 GOTO 1380
1370 GOTO 1450
1380 GOSUB 2060
1390 PRINT (NO)G,"AND CENTS ";
1400 LET Y=.01
1410 GOSUB 2060
1420 GOSUB 1650
1430 GOSUB 2060
1440 IF (X1-(X2(AS)10))/100(GT)=.01 THEN X=(X1-(X2(AS)10))/10
     0:GOTO 1400
1450 PRINT (NO)G,"ONLY."
1460 DIGITS= 0
1470 LET X=X9
1480 GOTO 420
1490 LET W$=STR$(X(K)):T2=X(K)/1000:T3=INT(T2):DIGITS=0:T$=STR$(
```

```
    T3)
1500 DIGITS= 2:Z£=RIGHT£(W£,6)
1510 PRINT (NO)G,'' BRAND RADIO.''; TAB(48); ''@US''; V(K); TAB(63-T); ''
    US''; T£;'', ''; Z£;
1520 GOTO 630
1530 LET W£=STR£(X):T2=X/1000:T3=INT(T2):DIGITS=0:T£=STR£(T3)
1540 DIGITS= 0:Z£=RIGHT£(W£,6)
1550 PRINT (NO)G,TAB(48);''TOTAL: ''; TAB(63-T);''US ''; T£;'', ''; Z£;
1560 RETURN
1570 LET W£=STR£(D1):T2=D1/1000:T3=INT(T2):DIGITS=0:T£=STR£(T3)
1580 DIGITS= 2:Z£=RIGHT£(W£,6)
1590 PRINT (NO)G,TAB(63-T);''US ''; T£;'', ''; Z£;
1600 GOTO 990
1610 LET W£=STR£(X):T2=X/1000:T3=INT(T2):DIGITS=0:T£=STR£(T3)
1620 DIGITS= 2:Z£=RIGHT£(W£,6)
1630 PRINT (NO)G,TAB(48) ''TOTAL:''; TAB(63-T); ''US ''; T£;'', '' Z£;
1640 GOTO 1020
1650 LET X1=X/Y:X2=INT(X1)
1660 IF X2(GT)=10 GOTO 1790
1670 GOSUB 2060
1680 ON X2 GOSUB 1700,1710,1720,1730,1740,1750,1760,1770,1780
1690 RETURN
1700 PRINT (NO)G,''ONE ''; :RETURN
1710 PRINT (NO)G,''TWO ''; :RETURN
1720 PRINT (NO)G,''THREE ''; :RETURN
1730 PRINT (NO)G,''FOUR ''; :RETURN
1740 PRINT (NO)G,'FIVE ''; :RETURN
1750 PRINT (NO)G,''SIX ''; :RETURN
1760 PRINT (NO)G,''SEVEN ''; :RETURN
1770 PRINT (NO)G,''EIGHT ''; :RETURN
1780 PRINT (NO)G,''NINE ''; :RETURN
1790 IF X2(GT)=20 GOTO 1830
1800 GOSUB 2060
1810 ON X2-9 GOSUB 1880,1890,1900,1910,1920,1930,1940,1950,1960
1820 RETURN
1830 LET X3=INT(X1/10)
1840 GOSUB 2060
1850 ON X3-1 GOSUB 1980,1990,2000,2010,2020,2030,2040,2050
1860 IF X2-(X3(AS)10)(LT)(GT)0 THEN X2=X3
1870 RETURN
1880 PRINT (NO)G,''TEN ''; :RETURN
1890 PRINT (NO)G,''ELEVEN ''; :RETURN
1900 PRINT (NO)G,''TWELVE ''; :RETURN
1910 PRINT (NO)G,''THIRTEEN ''; :RETURN
1920 PRINT (NO)G,''FOURTEEN ''; :RETURN
1930 PRINT (NO)G,''FIFTEEN ''; :RETURN
1940 PRINT (NO)G,''SIXTEEN ''; :RETURN
1950 PRINT (NO)G,''SEVENTEEN ''; :RETURN
1960 PRINT (NO)G,''EIGHTEEN ''; :RETURN
1970 PRINT (NO)G,''NINETEEN ''; :RETURN
1980 PRINT (NO)G,''TWENTY ''; :RETURN
1990 PRINT (NO)G,''THIRTY ''; :RETURN
```

```
2000 PRINT (NO)G,"FORTY ";:RETURN
2010 PRINT (NO)G,"FIFTY ";:RETURN
2020 PRINT (NO)G,"SIXTY ";:RETURN
2030 PRINT (NO)G,"SEVENTY ";:RETURN
2040 PRINT (NO)G,"EIGHTY ";:RETURN
2050 PRINT (NO)G,"NINETY ";:RETURN
2060 IF POS(LT)59 GOTO 2080
2070 PRINT (NO)G:PRINT(NO)G,"                ";
2080 RETURN
2090 END
```

(NO) = # (LT) = < (GT) = > (AS) = *

£ = #

We knew it was coming...

San Francisco Progress June 11, 1980

They call them "Compu-killers." These are the folks that have had it up to here with the computer monsters---and they strike back with anything they can get their hands on.

Sooner or later it was bound to happen. Being surrounded with all this modern day computerology, and with machines taking what could be described as an arrogant attitude, computercide is becoming a growing phenomenon.

According to computer expert Gary W. Dickson, in a recent magazine article, computer murder is becoming quite common all across the country.

We heard of an insurance salesman in New York a few days ago who became so furious with his computer and the errors it was sending back to him, he attacked it with a screwdriver.

The story is told of a California sheriff who fired a shot at his computer when it poured out the wrong arrests records.

Someone we heard about the other day poured honey into a computer---the only weapon that was near at hand.

Dickson says the University of Minnesota and MIT are very much concerned with this outbreak of computer murder. Both are starting a course for managers to teach them how to cope with employees who just can't take computers any more.

If this computer we are using now for this story says, "Wrong Command" just one more time, we'll sign up for the course, too----or grab a bottle of honey.